NoteIt

Computer Science Senior Project

Gurjeevan Bains, Waylin Wang

Advisor: Dr. Lubomir Stanchev

Winter 2017, Fall 2017

Table of Contents

Abstract

Miscommunication is a common struggle that many students face in today's world, despite the significant technological progress that we have made. With NoteIt, we aimed to solve this issue, by allowing students to join groups based off of the courses that they are enrolled in during the current quarter. Within the app, they can send and receive messages and events from everyone else in the group, or post private reminders to themselves. Communication can range from assignment clarification to exam preparation tips. Overall the development of the application went relatively well, despite some roadblocks along the way. We were satisfied with the progress we made and plan on continuing to work on it after the quarter ends, by adding even more functionality to it.

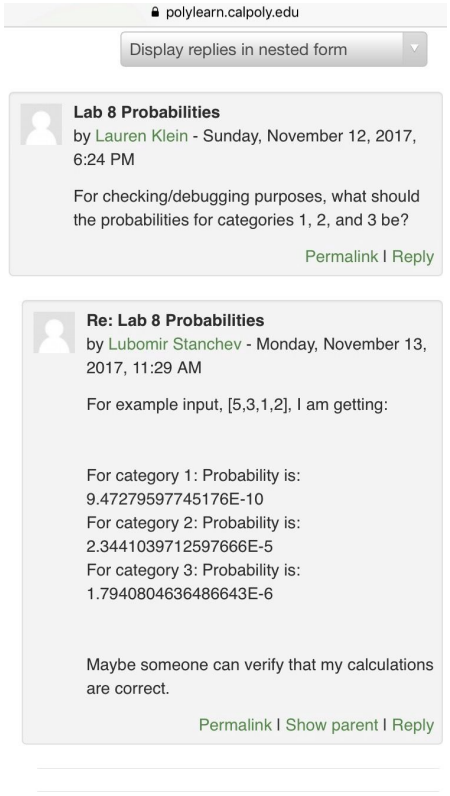
Background/Motivation

For our senior project, we are going to create an IOS application that is at its core, a combination of a community based reminder system and class forum. There is currently no application on the marketplace that allows students to interact with one another based off of the courses that they are enrolled in, while also serving as a personal reminder system. We believe

that the whole idea of using separate applications for setting reminders and interacting with classmates is inefficient.

In addition, most students have between zero and two contacts in any given course that they are enrolled in. Assuming they have a common question, those students would be out of luck, with no other contacts in the course. However, according to PsychCentral, college students spend about “eight to ten hours per day” on their cell phones. By being able to receive help and get questions answered directly on their smartphones, students will be less likely to forget about their homework assignments, and more likely to understand assignment specifications correctly. In addition, we plan on integrating a “community” around each course being offered. By this we mean that we plan to have different group style chats based off of the courses that students are currently enrolled in. This would help eliminate the awkwardness students may feel when asking for a random classmate’s contact information during the first few days of the course.

We understand that this is already available on polylearn, and on sites such as Piazza, but we believe that with a dedicated application such as the one we are proposing, it will get much more use than forums on polylearn, as it is much quicker to use. In fact, we conducted a survey of 30 Cal Poly students and found that only 3 out of the 30 students (10%) ever even used the polylearn forum page.



This image depicts the polylearn forum

When we asked the same group of 30 students whether they would use this mobile application, an astounding 26 out of the 30 students (87%) said that they would indeed use it. A mobile application would be much easier to use, as a question could be asked to the entire class with a few taps. We believe that these two features together would provide an incredible advantage to almost any student. Piazza actually offers a mobile application, but it has very mediocre reviews, with an average of 2.7/5. Some of the complaints include, crashing on login, and text that is too small for those with imperfect eyesight to view. We are definitely going to take these complaints with Piazza into account when building our app.

Really when it comes down to it, we wanted our senior project to be directly beneficial to our peers at Cal Poly. We did not want to simply mimic an already existing app, but create something of value that we could be proud of; something Cal Poly students could still be using 5 to 10 years down the line, well after our graduation date. There is absolutely nothing that offers this combination of features on the market, which is what will make the development of this app so exciting and challenging.

Introduction

We decided to go ahead and develop our application on the IOS platform. We spent a considerable amount of time debating between developing the application on IOS versus Android, and after doing some research and taking our own personal motives for creating the application into consideration, felt that this was the best choice. According to a study done by Nationwide, 53% of users in the United States use Android phones with only 44% using iPhones. However, amongst college aged students, 65% use iPhones, with only 35% using Android phones. Despite Android holding an edge with the US market as a whole, we felt that since our application was going to be used by college students and possibly high school students, we would capture a larger audience by developing the application on IOS.

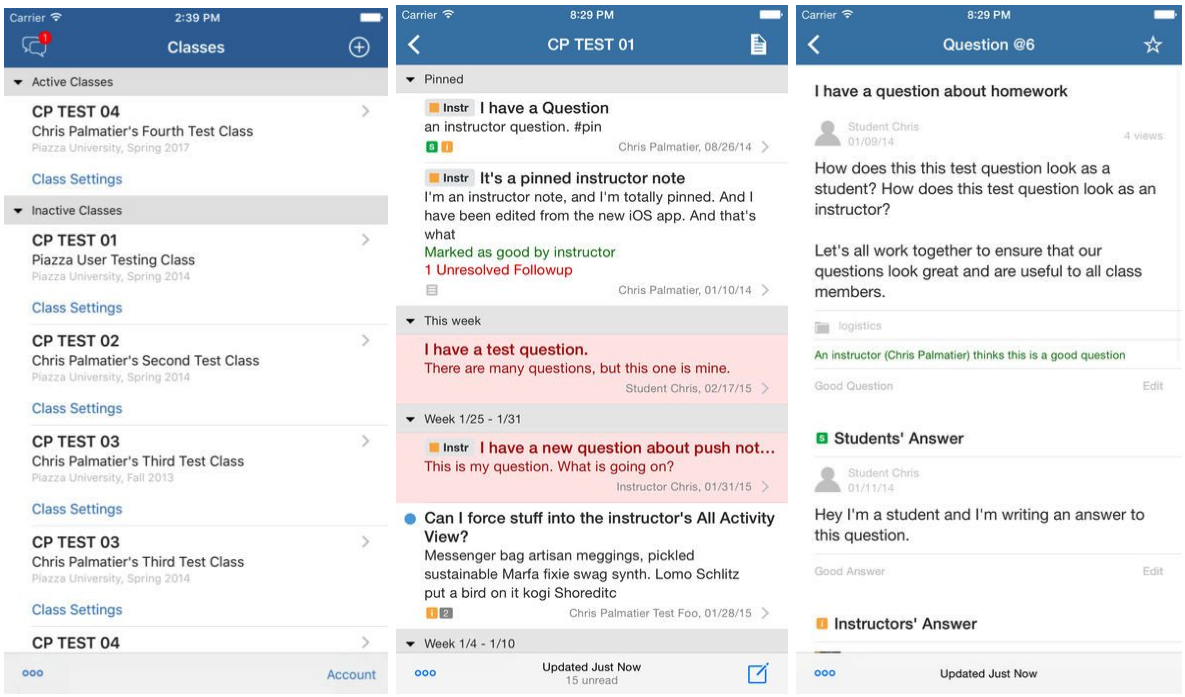
This is not to say that we did not also consider developing an Android application in conjunction with it. We thought that instead of fully developing the application on one platform, we could get a start on both platforms simultaneously, to capture a larger market right off the bat, but do partial implementations. Ultimately we decided against doing so, as we realized that

having a smaller initial test group was not too big of a downside. We also came to the conclusion that we would rather have it working on one platform completely rather than partially on multiple platforms. Ultimately, the Android implementation is something we will complete at a later date.

As far as deliverables go, there are several that we have planned, to ensure that the project is being completed in a timely manner. Since neither of us have any mobile application development experience, our first deliverable will be a simple home screen, that asks for the student's login credentials, or allows them to register for the app.

The next big deliverable will adding the current quarter's class offerings to the app so that students will be able to select which courses they are enrolled in. This may be somewhat difficult, as we will not be able to get a student's course information directly from the Cal Poly site, so we will have to find some other way to get this information. All of this will be added to a NoSQL database and will be updated once at the beginning of each quarter.

After that, we plan on creating a class listings page where students will be able to view the courses that they are currently enrolled in. Then we will create the events screen, where students will be able to view all of the events that have been sent for a particular course, and add their own events at well. At this point, students will be able to begin communication with each other, by clicking on a particular event and adding replies to it.



These images depict the basic architecture of the Piazza application, which we feel we can draw inspiration from.

This is just a rough outline of what we plan on doing. As a disclaimer, we may decide to add new features, modify existing ones, or remove them altogether if we decide that there is a better alternative available.

User Stories

1. As a student, I want to be able to ask questions about assignments and class logistics, and receive responses in the form of replies, so that when I won't be confused or misinformed.

2. As a student, I want to have the ability to add and delete courses, to account for courses that I have dropped and added after the quarter has begun.
3. As an instructor, I want to be able to send out reminders about class changes in a more speedy fashion than email, so that my students will be more quickly informed of changes.
4. As a student, I want to be able to give tips to other students on things I found useful in completing assignments, so that no other student is stuck feeling lost in the course.
5. As a student, I want to be able to create an account and register, so that my preferences and courses will be saved.
6. As a student, I want the add courses screen to feature the current quarter's courses, so that I can continue using the application, quarter after quarter.
7. As an instructor, I want events to have a student's user ID attached to it, so that I can address the student in private via email, if there is need for further discussion.
8. As an instructor, I want the add courses screen to give the ability to enroll for different sections of the course, so that I can be sure to address the different sections of my courses appropriately.
9. As a student, I want the events screen to list events listed from soonest to furthest due date.
10. As a student, I want the ability to make events public or private
11. As a student, I want the ability to delete events that I have created.
12. As a student and instructor, I want the events screen to refresh in real time so that I do not have to reload the screen myself when I suspect that events have changed.
13. As a student and instructor, I want the ability to edit events that I have created.

Swift Vs. Objective C: Our Choice

Upon deciding to develop our application on the IOS platform, we had a decision to make in regards to which programming language we would develop it in. On one hand, there was Objective C, which is the language that has existed since the dawn of IOS development, and on the other hand there was Swift, a new language that Apple was pushing for IOS developers to begin using.

After doing some research we found that there were several advantages that both Swift and Objective C had over one another. For starters, Swift runs more quickly than Objective C, and in fact, faster than C++. In addition, Swift uses the LLVM compiler, which is used by modern languages such as Python and Scala, and is more friendly to develop on. Swift also got rid of the use of pointers, which also makes it “safer” to develop on. Unlike Objective C, Swift supports Automatic Reference Counting, which helps to prevent memory leaks, and allows to the developer to be more productive. On the other hand, Objective C, being much more mature, has many more established conventions than Swift, being only a few years old. One could also argue that Objective C is more friendly to learn, with the myriad of tutorials available online for it.

With all this said, we decided that Swift would be a better choice for developing our application. We felt the advantages outweighed the downsides, even with its steeper learning curve. Since our main goal for our senior project was to learn more about IOS development we felt that learning Swift would be a good investment even if it means getting less work done

overall. The skillset it'd provide us with would be more beneficial than the extra work we'd get done with Objective C.

Firebase Background

We had several choices to choose from when deciding how to host our application's database and handle user authentication. Ultimately we found that Firebase provided us with a wide variety of features that would not only come in handy in the near term, but also as our app scales to thousands of users. We also considered going with AWS mobile, but felt that it was geared more towards larger enterprise mobile apps rather than ones on a smaller scale like ours.

One of the biggest reasons we chose to go with a third party hosting platform rather than our own infrastructure was because of scaling and efficiency. We simply do not know how quickly this app will take off, and having to order and maintain hardware because we chose to host the site on our own would not serve as the best use of our resources. In addition, we found that Firebase had everything that we needed and more, for a fraction of the initial cost that it would be for us to create our own small datacenter, as its payment system is based on usage.

For example, its real time database gives our application a modern backend and does much of our work for us in that we do not have to worry about sending requests to refresh data. In addition, this database is NoSQL in nature, a technology that is more ideal when scalability is a concern, as well as when data schemas are changing regularly. We were also very pleased with the analytics that Firebase provides, which we feel will be of great use at a later date; with it we will be able to better gauge where to place more of an emphasis in terms of bettering our

application in order to provide the best value for students and professors. This ties in with its crash reporting system, which can be used to group together users who face similar issues in order to send automated notifications to those users, or get a better idea of the demographics of those affected. It also features a robust authentication system that would otherwise take up to months to develop on our own. This authentication system features the same security used in Google Sign-In, meaning that we as developers can leverage the manpower of thousands of Google employees instead of using our own time and resources. Overall, we found that Firebase would allow us as developers to focus on development of new features rather than having to manage infrastructure. Especially for the short amount of time that we had to develop this application, this was its selling point. Had we hosted our own server and developed our own authentication system, we would not have completed nearly as much as we were able to do by using Firebase.

Quarter 1 Goals

- Learn the basics of Swift development in order to get a better understanding of what we will be able to accomplish within our timeline
- Decide on whether we will host the application ourselves or use a third party hosting platform
- Get an API token from Cal Poly IT in order to allow students to login to our app with their Cal Poly credentials

- Complete a detailed list of user stories and UI diagrams that will give us a good idea about how our application will look and behave
- Complete work on the registration and login screen
- Have the schema for our database completed
- Add in capability to list the courses that a logged in user has enrolled in
- Decide on an icon for the application as well as a name for it
- Add the ability to remove courses from the application that you are personally not interested in being in groups for
- Begin work on the calendar screen, which we are still deciding on an appropriate layout for

Quarter 1 Progress Report

During the first quarter of our senior project, we have met all of the goals that we set forth at the beginning of the quarter. Our main goals during these past few months were to test the waters, in the sense that we wanted to make sure that we had the pieces necessary in order to create this application. For one, we needed to be sure that we had access to either students' portals or a listing of all courses. Our original goal was to have students sign into the application with their Cal Poly credentials, but we realized that doing so was not possible without months of planning and approvals that we did not have the time to go through for the project. So we proceeded with instead getting a listing of all courses offered (updated quarterly), and instead using that to get the data that students need to add the courses they are enrolled in.

We also got our infrastructure intact during this quarter. We decided to use firebase to host our app, and proceeded to set up user authentication, and got the database set up for it. We then made UI sketches for what we want the various screens of the application to look like. After that we created the use cases for the application that we will continue to update during our next quarter.

At this point, we began to actually work on the application itself. This entailed first creating the login screen, and then working on allowing users to add courses they were enrolled in. These changes were then placed into the real time NoSQL database that is hosted on Firebase.

Thus far there have not really been any changes to our plans for the application except for the fact that we are not using Cal Poly authentication for user accounts. We do not anticipate any future changes to our final goal for the application at this point either, but this is of course subject to change.

So far, Gurjeevan has worked on the parsing script to get all of the data from schedules.calpoly.edu and into the Firebase Database. He also worked on highlighting of courses on the course adder page and the pushing of this data onto the Firebase Database. Waylin worked on setting up our Firebase instance, the user registration/login, and on displaying courses on the course adder page from the firebase database. We believe that as a team we have worked well.

Our plan for when we work on the project next remains the same. We still want this app to be adapted by Cal Poly campuswide. This is what would make the app most useful to all. With each student using it, everyone with the app will benefit. This will get the most number of students posting reminders and tips to the class, and this is what will allow the most students to

give helpful responses to otherx. We believe that by adapting this application, grades will increase for all Cal Poly students.

We will continue with the same pace that we have used during this quarter, as it has led to us meeting our goals thus far, so we see no reason to change that. We believe that next quarter we will make more progress early on during the quarter as well, as much of the administrative and infrastructure setup of the application has been completed. For more details on the end goal of the project you can review the proposal that we submitted early on during the quarter. As far as communication went with our professor, we found that everything went very well. We were in nearly constant communication, and met roughly once every 1-2 weeks, which we found to be sufficient. Professor Stanchev was helpful in providing us guidance in how to proceed when we needed it and provided helpful suggestions for features that would work well.

Quarter 2 Goals

Week of September 24-30: This week we will review what we already have done, and look over and revise UI diagrams if necessary (busy with career fair)

Week of October 1-7: This week we will update user stories and update parsing scripts if necessary because of changes to schedules.calpoly.edu site

Week of October 8-14: This week we will reconnect application to firebase; re-populate database with new data; we will also refamiliarize ourselves with the codebase

Week of October 15-21: This week we will create new screen “my classes” that shows a listing of all of the courses that a student is enrolled in. From this screen a student should be able to select any course which should display a new screen

Week of October 22-28: This week we will add the class view screen; for now the screen will be blank but will have a button that allows users to add a new comment to the class discussion page

Week of October 29-November 4: This week we will build up more of the class view screen; this screen will contain a list of messages that users have posted in the class discussion page; discussions could be anything from reminders, questions, comments, etc that everyone else enrolled in the class will see

Week of November 5-11: No meeting this week, away for veterans day but work on final writeup, test out application (share it with friends and ask for opinions/feedback we will include in writeup), catch up, and other last minute things

Week of November 12-18: This week we will work on final writeup and any other final last minute things/catch up

Week of November 19-25: Thanksgiving Holiday

Week of November 26-December 2: This week we will work on final writeup and any other final things/catch up

Weekly Progress

Quarter 1 (Winter 2017)

Week of January 8-14 - During this week we decided to become partners for the project and began asking professors whether they would be willing to advise us on the project.

Week of January 15-21 - During this week we met with Professor Stanchev for the first time and he told us to come up with an appropriate idea for an app. We knew that we wanted to develop an IOS application but were not yet sure what exactly we wanted to develop.

Week of January 22-28 - During this week we came up with the Cal Poly Reminders App idea. We discussed the application with Professor Stanchev who told us to come to him with a rough outline of what we wanted done in the form of a proposal. We found this somewhat difficult as neither of us had any experience with IOS development, and were not sure how difficult some of the features we wanted to include in the application would be.

Week of January 29-February 4 - During this Week we showed our finished proposal to Professor Stanchev. He approved it and told us to work on having user stories completed by the

next week. For this we spoke with several of our friends in order to get their feedback on what they would want in the application. We also spoke with Cal Poly IT in order to try and get an API token so that we could allow students to login to the application using their Cal Poly credentials. We found that this was particularly arduous, and required approvals which could take several weeks to omplete, which we did not have time for.

Week of February 5-11 - During this week we showed our completed user stories to Professor Stanchev who approved of them. At this point we told Professor Stanchev that during this week we wanted to do UI sketches for what we wanted the various screens of the application to look like. He agreed and told us to get them done by the next week. This was rather difficult as neither of us had much experience in designing UI's, and did not know how to make it both user friendly and simple to develop given our experience.

Week of February 12-18 - During this week we showed our sketches to Professor Stanchev. He approved of them, and we told him that we were ready to begin development of the application. So this week we did some exploring on possible infrastructure choices for our application. This was particularly difficult due to the wide range of choices available, but we eventually settled on Firebase, as we liked the fact that it was a NoSql Database, real time in nature, and took care of user authentication for us.

Week of February 19-25 - During this week we explained our infrastructure choices to Professor Stanchev, who approved of them. He told us to come up with a database schema, which we did,

and told us to get the database up by the next week. So we were able to accomplish this, but realized that without Cal Poly authentication, we would have to get our courses from elsewhere, which is when we found out that all courses can be found on *schedules.calpoly.edu*. What we found difficult about our tasks this week were getting familiar with how to design NoSQL databases as our experiences lied mainly in relational databases.

Week of February 26-March 4 - During this week we informed Professor Stanchev about the discovery we made regarding accessing courses. He approved of our idea of getting the courses from *schedules.calpoly.edu.* So this week we worked on creating a parsing script that would create a json file with all of the relevant information for each course in it, so that we could easily upload it onto the Firebase noSQL database. What made this difficult was lots of inconsistencies within the data, which we had to account for.

Week of March 5-11 - During this week we showed Professor Stanchev the progress that we made with the database. We showed him all of the courses and sections put into the firebase database, which he thought we did superbly. So this week we worked on creating the login screen for the application. The login is connected through firebase as well, which does all of the authenticating for us. We also worked on creating the screen that displays all of the courses available for adding. The difficulties this week were largely due to our inexperiences with Swift development.

Week of March 12-18 - We showed Professor Stanchev the progress with our application this week and he was pleased with the results. He requested that we finalize everything we wanted to have done for the week before Friday and complete a progress document. So during this week we worked on a progress document to show to Professor Stanchev as well as finalizing what we wanted to get done to show Professor Stanchev for the quarter. This includes behavior allowing students to create an account on the application and be able to add classes that they are enrolled in. We packaged it all neatly in a zip file and gave it to Professor Stanchev. As far as difficulties go this week, there was nothing out of the ordinary. Most of the work was putting finishing touches on what we wanted to have done by the end of the quarter.

Quarter 2 (Fall 2017)

Week of September 24-30 - During this week we finalized our work on adding the ability for students to select courses that they are enrolled in, and have them added to the student's account. We showed this to Professor Stanchev, who was happy with our progress. We did encounter some difficulties in developing this because of strange errors we encountered with section highlighting. We also changed the layout of the database this week, which meant changing the parsing script and re-uploading the data into firebase. Professor Stanchev told us to have a list of goals completed by the next week.

Week of October 1-7 - This week we completed the set of goals that Professor Stanchev told us to have done. Again, the difficulty in this lies in us not knowing how much we would be able to

have done given our limited experience with IOS development. So we set some rough goals with room for error, and planned to do our best to meet those goals and if time allows, go above and beyond. Professor Stanchev was satisfied with our goal projections and asked for us to complete the class listing screen by next week. This screen will show a listing of the courses that the currently logged in student is enrolled in.

Week of October 8-14 - This week we worked on the class listing screen. From here students can view all of the class that they are enrolled in. Professor Stanchev noted our progress on the screen. The screen should be completed by the next week. Fortunately we did not encounter many difficulties this week. We also worked on updating some of the UI sketches to model the updated way we want our application to look.

Week of October 15-21 - This week we completed work on the class listing screen. As per the goals we set, the next step in the development of the application is adding messaging functionality to the application. This will take several weeks to complete, as this is what a bulk of our remaining work will be. We did not meet with Professor Stanchev this week, but we completed what we were supposed to have done, so will show him during our next meeting. Again this week things went rather smoothly and there were no particular blockers that we encountered.

Week of October 22-28 - This week we added new features to the class listing screen such as removing of courses and adding new courses, without overwriting existing ones. We showed our

work to Professor Stanchev who was satisfied with what we completed. We also worked on the UI for the next two screen to complete, which are the events screen and add new event screen. We did not encounter any difficulties in completing the work this week.

Week of October 29-November 4 - During this week we worked on a partial completion of the event viewing screen. As of now, we are displaying events in order from soonest to latest due date, but are not yet able to show the events in sections with labeled due dates, which is the next thing that we have to complete. The biggest hurdle we had to overcome this week was working with Firebase queries, which we found difficult given our limited exposure to working with callback functions. We showed everything that we completed to Professor Stanchev, who was very happy with the work we had done.

Week of November 5-11 - During this week we completed work on the events screen. We added functionality to delete events, and also added sections to the app which makes it very easy for the user to see what he should focus his work on. We were unable to meet with Professor Stanchev this week, but gave him an idea of what we would be working on during our last meeting. We also continued work on the final report. In terms of blockers, there were a few problems that we encountered with the sections, because of the way that we had previously architected the events viewing screen. This required an overhaul of the queries that we previously used in ordered to fetch what we needed to view.

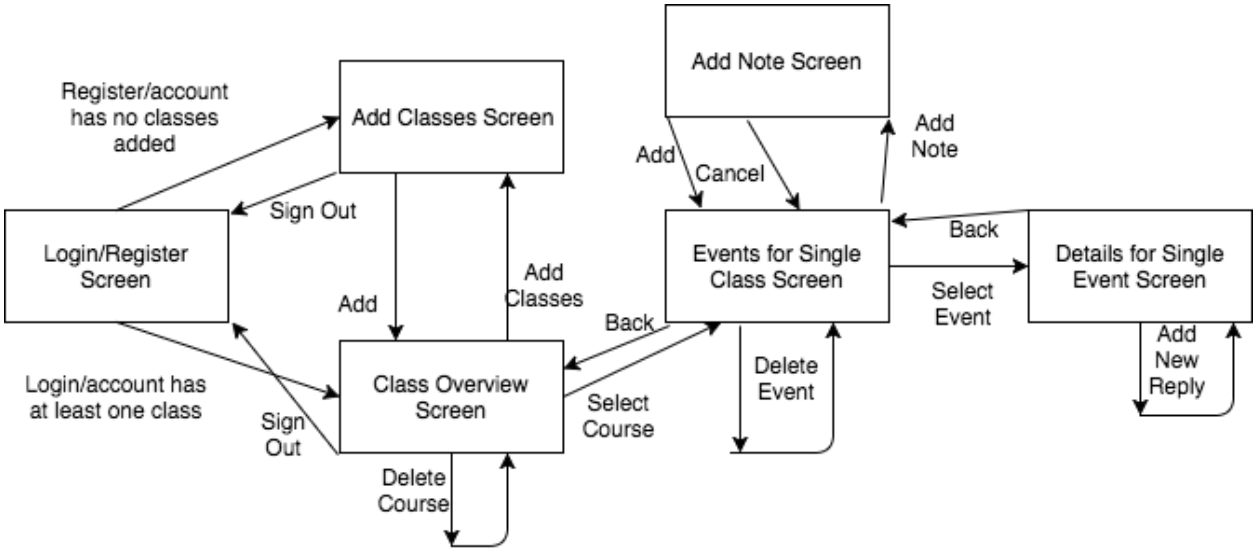
Week of November 12-18 - During this week we worked on the segue to the class forum, or replies screen, as well as a full implementation of that screen. We did not have many difficulties in completing this as most of the difficulty came from passing the appropriate data from the events screen. We showed our work to Professor Stanchev in the form of a (near) final demo, and he was very happy with our progress, and requested that we add a final feature to the app, which is editing of events. We also continued to work on the final report.

Week of November 19-25 - Thanksgiving Holiday. We did not work on the project during this week, but did work on the final report.

Week of November 26-December 2 - During this week we worked on editing existing notes and completing the report. We demoed our work to Professor Stanchev, who was happy with the final result, and we turned in our work to him and the Cal Poly Library.

Architecture

Before beginning a detailed discussion of how the entire application works and how the various views connect to one another, it may be helpful to spend some time with the following flow diagram, to get a basic understanding of the flow of the application.

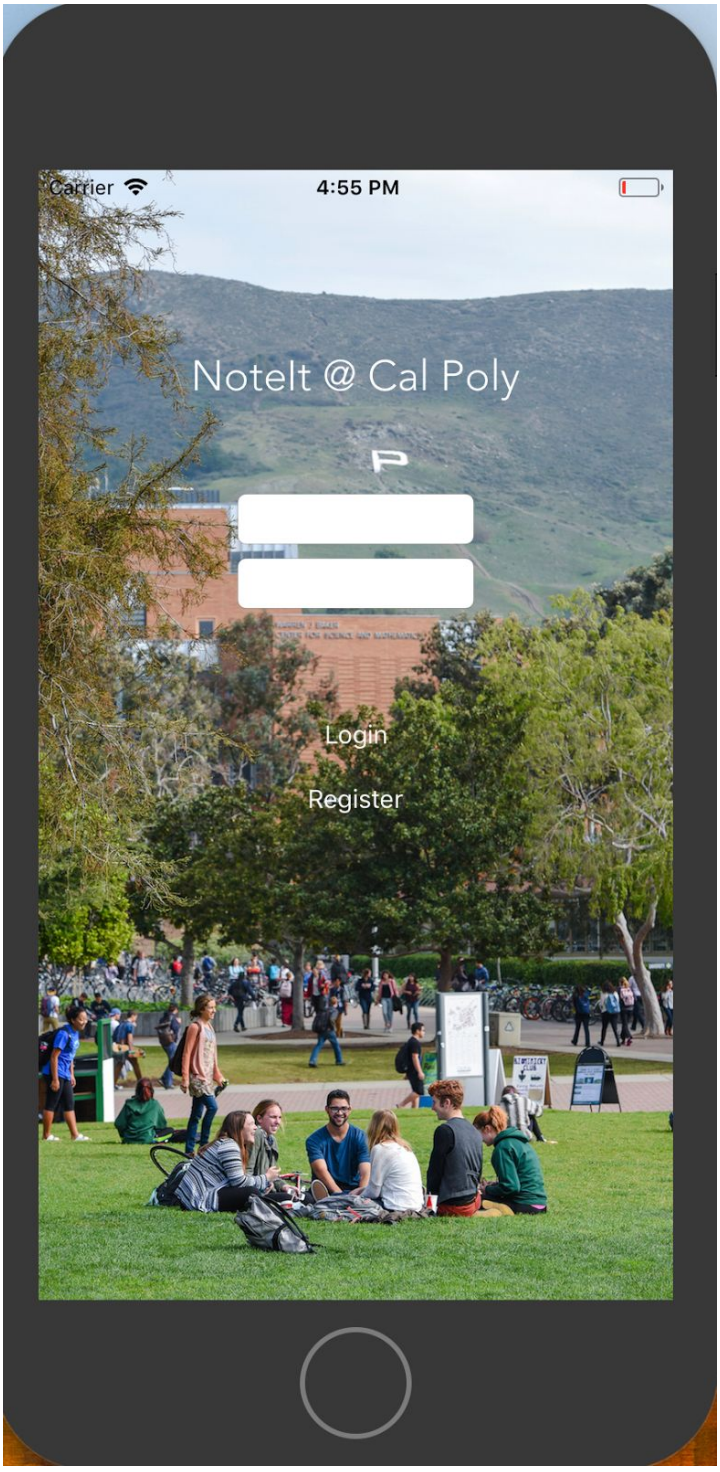


Simple Flow Diagram of the application

It could also be helpful to take a closer look at some of the structs that we created, as it will make understanding the application's architecture much easier. One such struct is the "User" struct, which simply contains two fields for user id and email. The "ClassItem" item struct has fields to represent the various attributes that can be used to represnt a course, such as department, time, instructor, section, and course number. The "Note" struct is what we used to represent events. Its attributes include a unique identifier, "id", a field describing whether it is public, a title, a body, a due date, a tag field, a user id field detailing who created the event, and a created date field detailing when the note was created. We decided not to include replies in the "Note" struct itself but instead make another struct "NoteItem" which would contain the "Note" along with an array of "Reply" objects. In addition, this "NoteItem" struct also has a field for latest text, which is used to display the first 40 characters of a Note's title. As far as the "Reply" struct

goes, it includes similar fields to Note, such as created date, id, and user id, and body. However, it also contains a field, parent id, that is used to tie it to the parent note that it is a reply to.

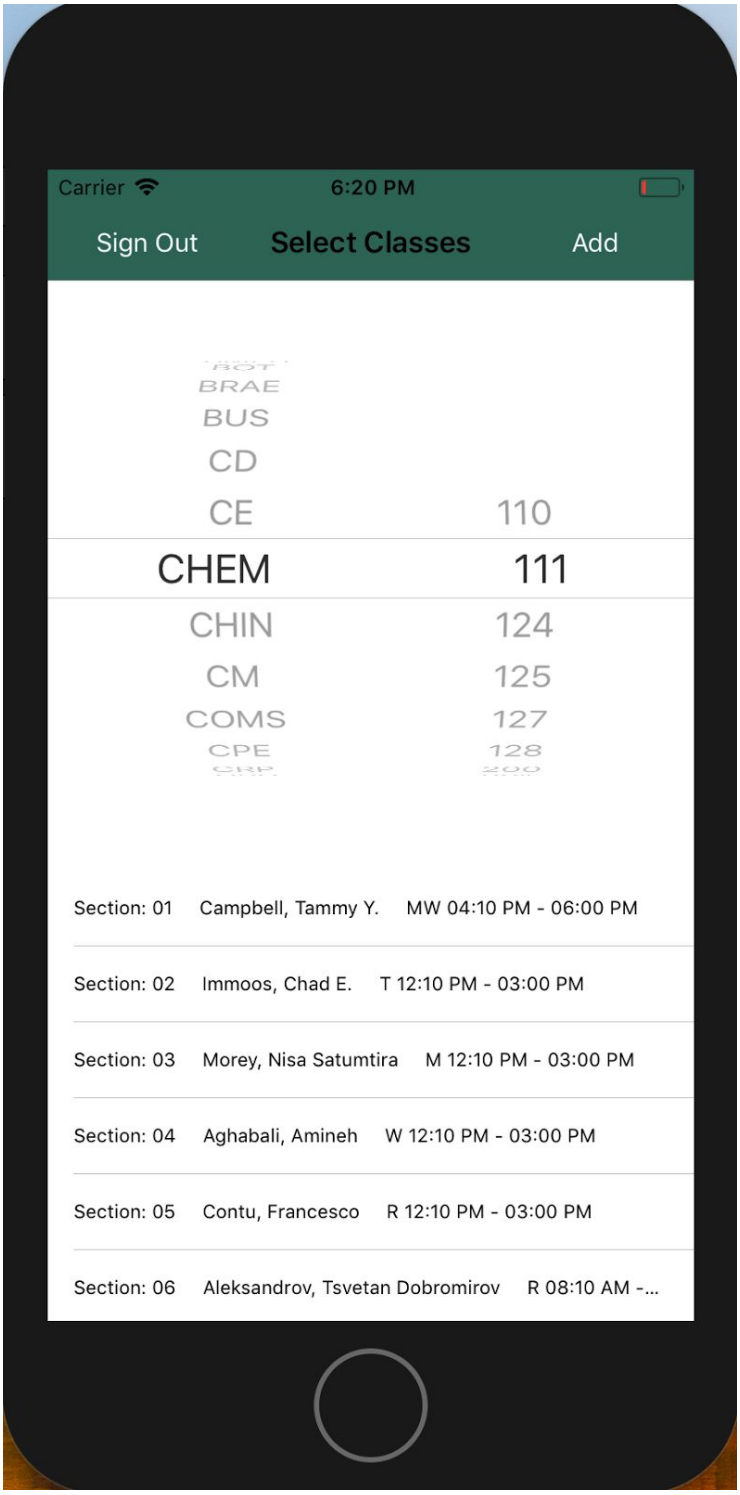
We will now examine the various views that the application has, and describe what is happening in the backend for each of them. The first view we will examine is the login screen.



NoteIt Login Screen

Before even loading the login screen, we determine whether a user is already logged in on the current device. Then depending on if the user has ever added a course to their profile, we take them to straight to either the add courses screen or overview screen. Only if no user is logged in is the login screen displayed. A listener is also attatched, so if the login state does change, then the login screen will be automatically segued to. Assuming a user is not signed in, the user will be greeted by the screen shown above. From here, if the user enters an email and password, a request will be fired to firebase with the supplied email and password. If the supplied fields match a user in the database, the user will be segued to either the add classes screen or overview screen. Selecting the register button will open up an alert window with fields to enter an email and password. Cancelling will take the user back to the login screen, but selecting the save option with an email and password entered will create a user with those credentials in firebase and log the user into the application.

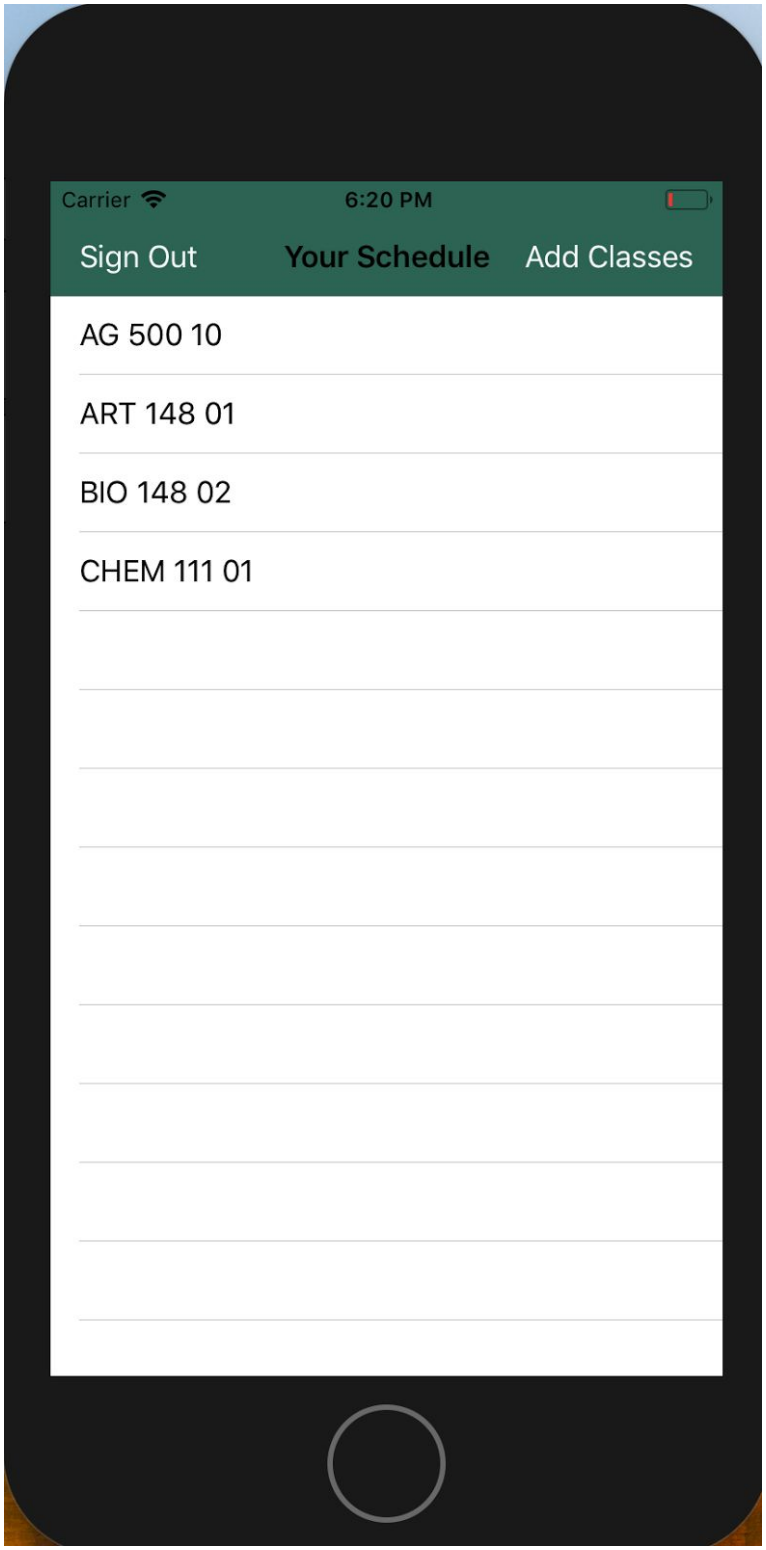
The next screen we will examine is the add classes screen.



NoteIt Add Classes Screen

The first thing that happens is a query that fetches all of the courses currently in the database. The courses are sorted by department and class number, and put into arrays so that the UIpickerview (shown in above image) can have the correct data to display to the user. A separate query is also run to take note of the courses that the logged in student has already enrolled in, so that they will not be selectable. At this point the picker view has loaded and the user can select a course by department and course number. Doing this will load the bottom half of the screen showing the sections available for the currently selected course. Selecting any of the sections available (provided you are not already enrolled in that section), will hilight it in a different color and add it to an array (selecting the section again will remove it from the array and return it to the original color). Upon selecting the "Add" button on the top right corner of the screen, the courses in the array will be added to firebase under the current user's id, and a segue will be performed to the class overview screen. Selecting the sign out button will log out the current user and return them to the login screen.

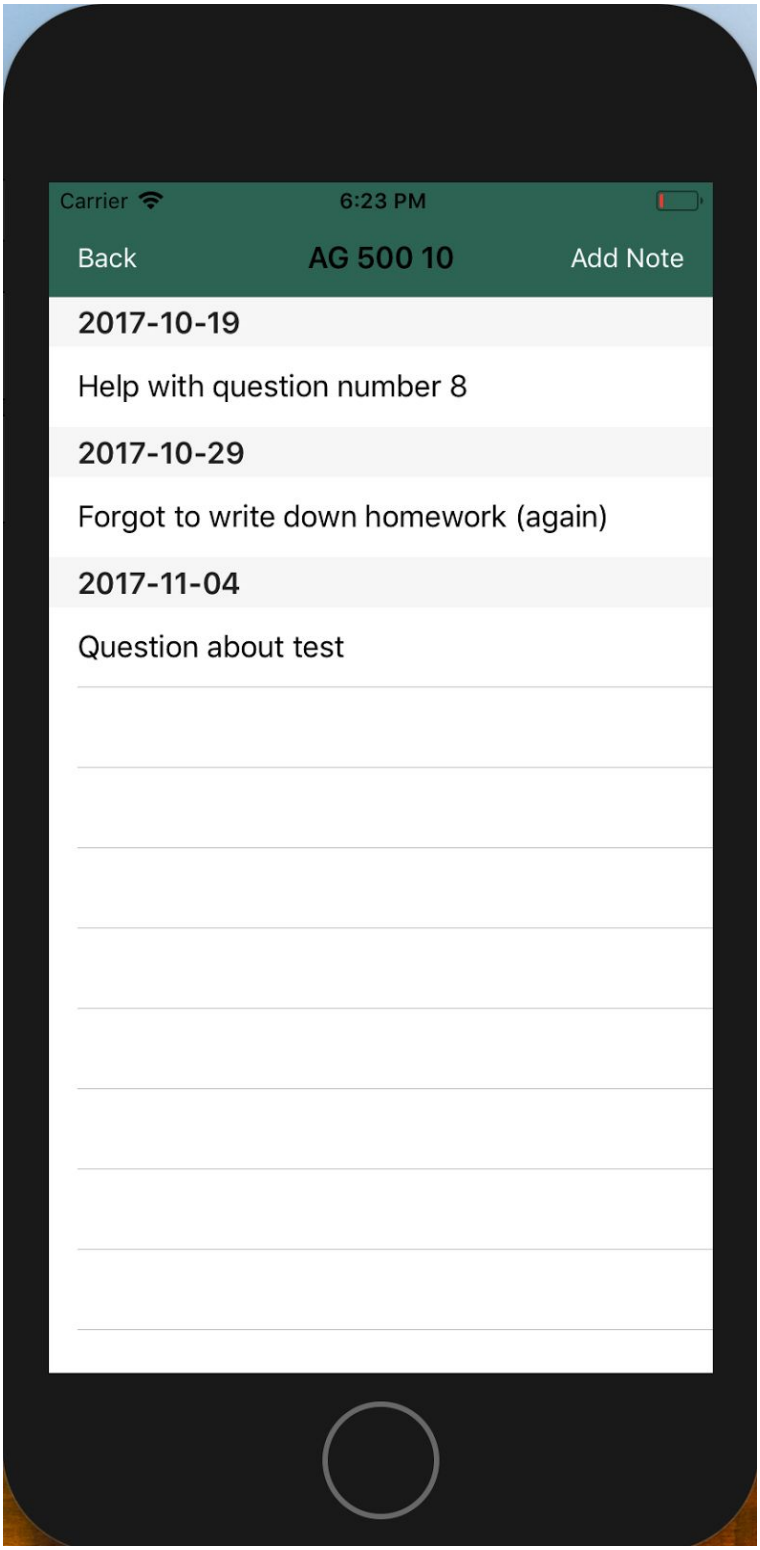
The next screen we will examine is the class overview screen.



NoteIt Overview Screen

Before the class overview screen loads, a query is run on the database, for all of the courses that the currently logged in student is enrolled in. These courses are added to an array for displaying purposes. The query also runs again each time an update is made to the logged in user's courses. Functionality has also been added for delete, so a left swipe on a particular cell gives the user the option to delete a course. Doing so triggers the query (which has been listening for updates to its table), and reloads the view. Selecting the sign out button will log out the current user and return them to the login screen. Selecting any particular cell will initiate a segue onto a screen that shows the events for a single course.

The next screen we will examine is the events for single class screen.

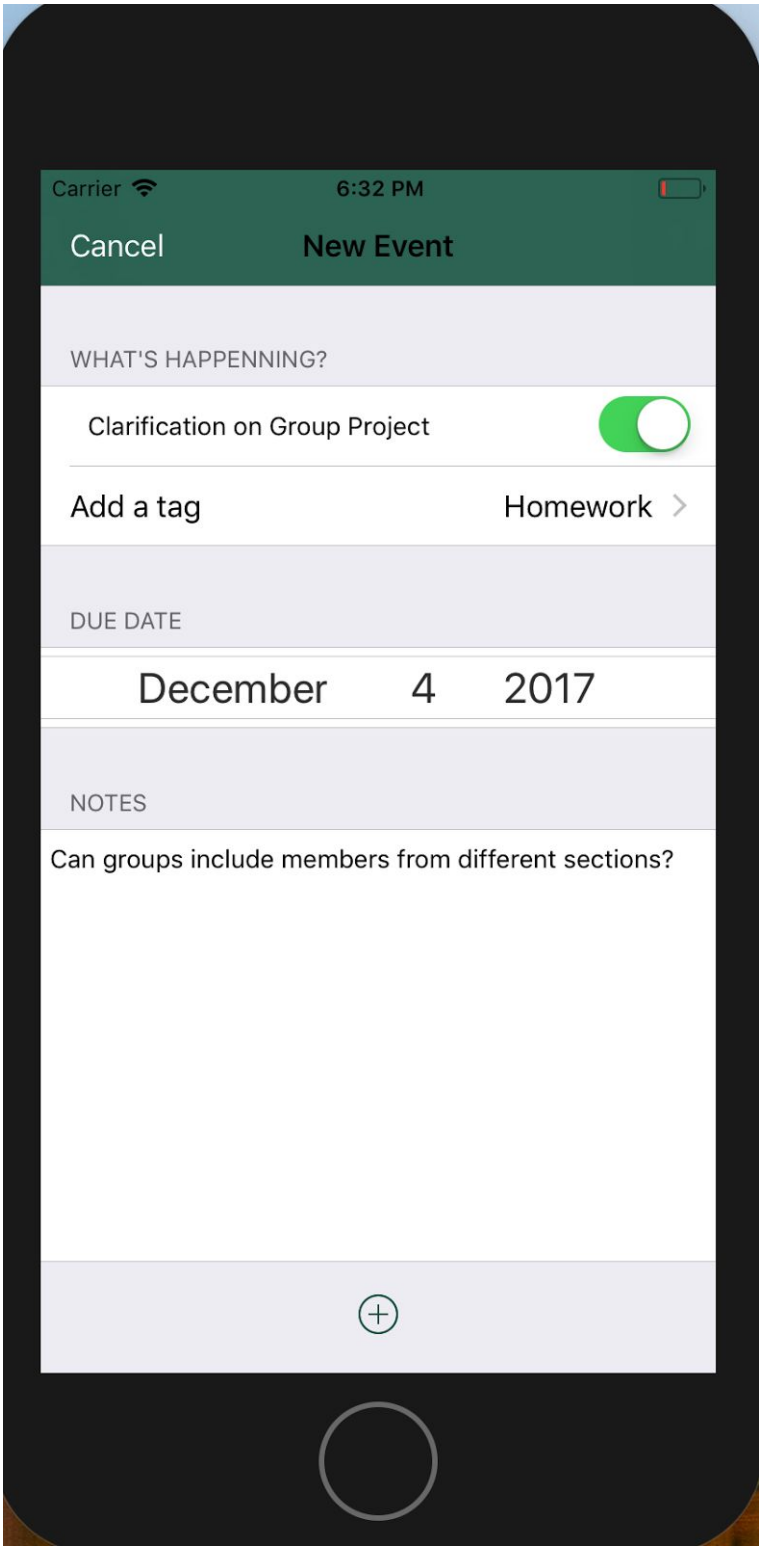


NoteIt Events For Single Class Screen

Before this screen loads, a query is run on the classnotes table of the database. For each of the classnotes that is returned, a check is run to ensure that the classnote is either available for public viewing, or belongs to the currently logged in user. For all classnotes that pass this check, the list of replies belonging to this note are added to an array. An additional query is then run on all of the appropriate note id's, to return the appropriate note from the notes table that corresponds to the id of the classnote (this note is then taken and used to create a noteitem object). At this point, the due date for the note is extracted and added to an array of due dates (only if the array does not already contain that due date). These unqiue due dates will be used for displaying the events in sections according to their due dates, with the closest date first. Next, another query is run in order to get the appropriate children for the current note (recall that we placed an array of children id's for each valid classnote into an array for this purpose). Each valid child is placed into the noteitem object, and at this point the noteitem is added to a dictionary that has due dates as keys, and arrays of noteitems with that due date as values. The query is run with a listener so that any change in the database causes a rerun of the entire callback, which is what will allow automatic updating of the events screen each time changes happen. The reason why we chose to use a dictionary to hold the noteitems instead of an array, was because of the fact that we were using sectioning to display events, and having an easy way to access all of the events that belong to a particular section (due date in our case) would be made much easier through the use of a dictionary. We also added the ability to delete events, which is triggered in the same fashion as deleting courses from a schedule (left swipe on the cell you want to delete). In the backend, the only difference is that we must delete data from 3 tables in the database rather than just one. We have to delete the classitem, note, and all replies

corresponding to the event from the replies table. Upon selection of a particular cell, a segue initiates to the event details screen, and the noteitem corresponding to the selected cell is sent, for purpose of displaying the event's contents as well as replies. Selecting the back button segues the user back to the overview screen, while selecting the "Add Note" button segues the user to the add event screen.

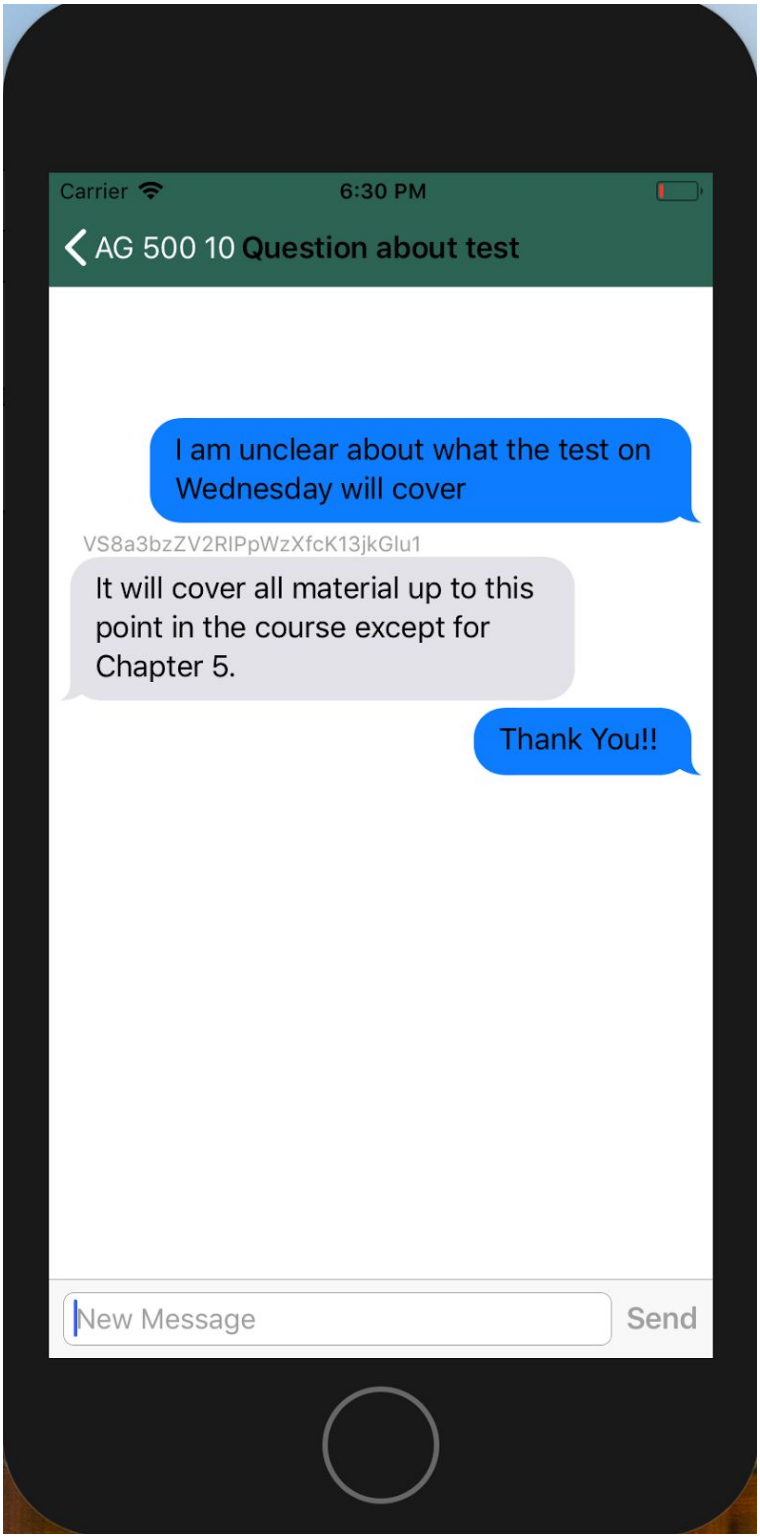
We will now take a closer look at the add event screen.



NoteIt Add Event Screen

Selecting the cancel button on this screen will segue the user back to the single class events screen. Selecting the “+” button will create a note object from the current data in the title, due date, body, private, and tag fields. It will also then add this note object to the notes table in the database as well as some of its data to the classnotes table. It then segues back to the single events table.

Next we will examine the details for single event screen.



NoteIt Event Details Screen

Selecting the button on the top left seges the user back to the single class events screen. For displaying the messages on this screen we used a third-party module called JSQMessage. Before this screen loads, the note body is loaded into an array of JSQMessages. Then, each of the replies are also added to the array. Note that since the replies were sorted in ascending order of creation date before being passed to this view controller, there is no sorting needed here in order to have the messages appear in order of creation date. A JSQMessage object has fields sender id, sender display name, date, and text. The messages array is then displayed, with messages appearing on the right or left sides of the screen depending on whether the message was created by the currently logged in user or some other user. Above each message not created by the currently logged in user is also that user's id. Selecting the text box at the bottom of the screen allows the user to add a new reply to the event. Selecting the send button will add the reply to the array of reply id's in the classnotes table in the database as well as to the replies table. It also displays the new reply to the screen.

Individual Contributions

During the course of the project, Gurjeevan wrote the required documentation that Professor Stanchev requested. This includes progress reports, proposals, and the final report. He also completed the parsing scripts to get the data from the schedules.calpoly.edu website. In terms of the actual application, he worked on highlighting of selected courses and sending them to firebase on the "Add classes" screen. He also worked on the "Class Events" screen, which

involved retrieving events from firebase, displaying them, and sending appropriate data to the detailed events viewing page on selection of a particular event.

Waylin took on more of the technical workload during the course of the project. He created the project's infrastructure, which involved setting up an account on Firebase, creating the database schema, and learning about how the application could interact with it. He also worked on the application's User Interface, which involved working on the application's storyboard. In addition, he worked on the "Login" screen, the displaying of available courses to add on the "Add Classes" screen, and also on the "Event Details" screen. The event details screen is where replies to events were made available for viewing, and additional replies could be sent.

Technical Difficulties Faced

Much of the difficulty that we faced in the development of this project came from our limited experience in working on mobile applications. On top of that, neither of us were at all familiar with Swift conventions, which took some time to get used to. In particular, we found working with the storyboard to be somewhat challenging. Simply setting up and connecting the various view controllers required us to spend considerable amounts of time researching how we could set them up to look and behave the way that we wanted. In addition to that, we had some trouble working with the firebase API. We found that there was not much support available on the web for it. In particular, something that took us quite a while to figure out was the fact that firebase queries get executed in separate background threads, which is why we were getting

some errors due to race conditions. However, overall, the process ran rather smoothly for the limited experience we had. Once we got more familiar with mobile application development conventions in Swift the development process ran relatively smoothly.

Conclusion

Overall, we are more than satisfied with what we were able to accomplish during the two quarters that we worked on the application. Not only did we meet each of the goals that we set forth to accomplish at the beginning of this quarter, but we exceeded them. Our goal for the end of the second quarter was to have a functioning message group for each class section. This did not include having “events” and then allowing replies, but a simple message board for each class where all activity would take place. However, as we continued to develop the application, we decided to exceed the goals we had previously set forth. Having “events” would make the app much more user friendly and add the potential for many future features as well. For one, it would allow us to add fields when creating new events such as due date, and whether an event was private or not. It would also make finding particular topics much easier than scrolling through possibly hundreds of messages that have been added since the beginning of the course. We also liked the benefits of events because they gave us the ability to sort them in accordance to their due dates.

There were also several other smaller features that we added to the application at a later date. For one, we added the ability to add additional classes to a student’s account, when previously, we only allowed replacing the courses that they had previously enrolled in with an

entirely new selection of courses. In addition, we allowed the removal of courses from a student's schedule with an easy to use left swipe gesture. We also added the ability to remove and edit events, but only those that the logged in user created. However, this is not to say that we consider this application to be complete by any means. See the "Future Work" section for more information on what we still have left to complete in terms of features.

There is not much that we feel that we could have done differently to the application as it stands now. Certainly, there were certain design decisions that we made early on and had to change as we got a better understanding of how we wanted the application to be architected; but in terms of the current state of the application, there is nothing that we feel should be different.

Working with Professor Stanchev was an absolute pleasure. He provided us with the right amount of guidance and support. We never felt that he was too overbearing, or not helpful enough. He offered advice when needed, and was very flexible with our meeting times. He was also very accommodating with the deadlines we set. He was happy to move them around provided we complete our initial proposed features by the end of the quarter. This was very generous on his part as there were certain weeks during which we had other priorities that got in the way of how much work we could complete. In summary, he allowed us to be independent in working on our application, has been excellent preparation for when we enter the workforce.

Future Goals

Though we have made significant progress on our application, there are still some features that we would have liked to implement but did not have the necessary time to complete.

We are planning on continuing to work on this application after the term completes on our own, as a personal project, because we are extremely passionate about it, and want to make it as useful as possible for our student body.

The first of our goals is to complete an identical android implementation of the application. We had originally planned on completing a partial android implementation alongside our IOS implementation, but ultimately decided to more thoroughly complete the IOS version before moving to a different platform. According to the Huffington Post, 96.7% of college students with smartphones have either an iPhone or Android phone, so by capturing these two platforms we will be able to service a majority of students.

Another one of our future goals is to differentiate between the various event types that we have created. As of now we allow the user to select which type event they want to create, but we do not display, or offer any additional options depending on which event type they have selected. For example, a reminder should give the user the option to add a due date, but a general note may not necessarily require one. One of the options that we have discussed adding include color coding the various event types and allowing filtering so that students can easily sort through them. In addition, we would like to add the ability to tag users in events and replies, which would then notify the tagged users and allow them to respond appropriately. This would then be tied to a push notification system that would notify users when either their message has been responded to, or when they have been tagged in a post.

We also want to add an additional viewing option to our application in the form of a calendar. As of now, we have our events listed in order of soonest to furthest due date in a list view. In our opinion, this is a much more effective use of screen real estate than a traditional

calendar, but we understand that not everyone shares this opinion. Some individuals prefer a traditional calendar view, which is why we would like to add the option to also view events in a calendar. Then, by clicking on a particular day, a user could view the events scheduled as "due" on that day, and click on a particular event to view responses.

This image depicts a calendar view that is similar to what we envision adding to NoteIt.

We are also planning on adding differentiation between student and instructor accounts. We could do this by changing the current login and registration system to instead use students' and instructors' Cal Poly credentials, via API tokens. We did not have the necessary time to complete this during the term because of the amount of paperwork that was needed in order to receive the API tokens, but doing so would definitely allow us to create admin and student accounts, which would provide much needed monitoring to the application. This way, if a

student posts something that the instructor deems as inappropriate, such as homework solutions, the instructor could remove the post.

In addition, we want to allow students to add file attachments to their messages. This could be useful if for example, a student asks for class notes for a day in which they were unable to attend. A student who did attend could quite easily send a picture of the class notes to the student in need. This would be a tremendous benefit to instructors as it would free up a lot of time that they have to spend answering emails about what was covered on a certain day.

Resources

https://www.cleveroad.com/blog/swift-vs-objective-c--what-is-the-best-language-for-ios-development-

https://www.forbes.com/sites/georgeanders/2016/10/04/android-or-iphone-teen-dilemma-sways-college-choices/#5453773d3317

https://piazza.com/

https://psychcentral.com/news/2014/08/31/new-study-finds-cell-phone-addiction-increasingly-realistic-possibility/74312.html

https://www.raizlabs.com/dev/2016/12/firebase-case-study/